## REMARKS

The above Amendments and these Remarks are in reply to the Office Action mailed November 14, 2007.

Currently, claims 1-49 are pending. Applicants have amended claims 1, 29, and 36 and added new claims 48 and 49. Applicants respectfully request reconsideration of claims 1-49.

I. Rejection of Claims 1-6, 8-9, 12-18, 20-22, 25-27, 29-34, 36-39, 41-46 Under 35 U.S.C. §102(b)

Claims 1-6, 8-9, 12-18, 20-22, 25-27, 29-34, 36-39, 41-46 have been rejected under 35 U.S.C. §102(b) as being anticipated by Chow (US 6,131,187). Applicants respectfully traverse the rejection because Chow does not disclose each and every limitation recited in claims 1-6, 8-9, 12-18, 20-22, 25-27, 29-34, 36-39, 41-46.

Claim 1 is not anticipated by Chow because Chow does not disclose "adding new code to said existing object code including adding object code for an additional method that adds an additional function and object code that provides said result to said additional method, said object code that provides said result to said additional method is added to said first method." Instead, Chow discloses a computer system that translates exception handling semantics for a "method" from object code to high-level code. When object code is received at a computer system from a server, the bytecode can be "translated into instructions of a compilable high-level software language" and is subsequently "compiled by an appropriate compiler into native executable modules" (Chow col. 1, lines 44-48). This allows the computer to process the object code in executable modules that are more familiar (i.e. "native") to the computer system. In order to translate the existing object code into a high-level language familiar to the computer system, an "empty bytecode information array is first established" (Chow Figure 2, step 41 and col. 4, lines 24-25). The array is filled "pertinent information… utilized for the translation of bytecode into a compilable high-level code such as C" (Chow col. 4, lines 31-34). This information is "obtained by stepping through each bytecode within a given bytecode stream" (Chow Figure 2, step 42 and col. 4, lines 54-56). The information can include a program counter, stack pointer, exception handler program counter, etc. (see Chow Table I). This step is

performed for the entire bytecode stream received from the server, including any bytecode for exception handlers. "The exception framelist is then obtained from a class file… [and] the entries in bytecode information array 35 that correspond to a starting location and an ending location of the exception framelist are marked" in the array (Chow Figure 2, steps 43-44 and col. 5, lines 2-10). The "compilable high-level code is generated based on the established bytecode information array" (Chow col. 5, lines 19-20). The high-level code is generated for the entire bytecode stream, including the bytecode for exception handlers. Additionally, labels referencing the array are added to the high-level code. For any code that "is able to generate an exception, [the system will] check if the exception occurs; if so, [it will] generate the high-level code to go to the label of the current [exception handler]" (Chow col. 5, lines 27-29).

The Examiner argues that the "new code" of claim 1 is the generated high-level code that leads to the label of the exception handler when an exception occurs, where the exception handler is the "additional method." The Examiner further argues that this "new code" is added to the "first method" that produces the exception to provide the exception to the "additional method," where the exception is the "result." However, this "new code" that is added to point to the label for the "additional method" does not also include "object code for [the] additional method." If the "additional method" is the exception handler, as the Examiner argues, the "additional method" is not part of the "new code" that is added to the "existing object code." Instead, the exception handler is part of the original code. Additionally, even assuming the "additional method" can be equated to the exception handler, this "additional method" does not add "an additional function" because the exception handler has the same "function" it did prior to the translation to high-level code. No new functionality is added by the "new code." Therefore, Applicants respectfully assert that claim 1 is not anticipated by Chow because Chow does not disclose "adding new code to said existing object code including adding object code for an additional method that adds an additional function and object code that provides said result to said additional method, said object code that provides said result to said additional method is added to said first method."

Furthermore, claim 1 is not anticipated by Chow because the "new code" in Chow is not added to "existing object code." Instead, the "new code" is high-level code added to the newly

generated high-level version of the "existing object code." Additionally, the "new code" that leads to the high-level code for the exception handler is not "object code." Instead, the "new code" is "high-level code [generated] to go to the label of the current [exception handler]" (Chow col. 5, lines 27-29). Therefore, Applicants respectfully assert that claim 1 is not anticipated by Chow for the reasons stated above. Claims 2-6, 8-9, 30-34, and 37-39 each contain a similar feature and are patentable over Chow for at least the same reasons as claim 1. Applicants respectfully request reconsideration of claims 1-6, 8-9, 30-34, and 37-39.

Additionally, claim 2 is not anticipated by Chow because Chow does not disclose that "said result includes a data item to be returned by said first method." The Examiner argues that the information used to fill the bytecode information array (i.e. pc, sp, eh, etc.) are all data items returned by executing the procedure. However, if the examiner argues that the "first method" is the bytecode producing the exception, this "first method" does not produce pc, sp, eh, etc. "when said first method is executed." This is merely information obtained by stepping through the bytecode stream. Therefore, claim 2 is not anticipated by Chow because Chow does not disclose that "said result includes a data item to be returned by said first method." Claims 12, 30, and 42 each contain a similar feature and are not anticipated by Chow for at least the same reasons as claim 2.

Additionally, claim 9 is not anticipated by Chow because Chow does not disclose "adding start byte code; adjusting byte code indices; adding exit byte code." The Examiner argues that the added "byte code" is the information contained in the bytecode information array (i.e. pc, sp, eh, etc.). However, this "byte code" is added to the array; it is not "new code" added to "said existing object code." Therefore, claim 9 is not anticipated by Chow because Chow does not disclose these features. Claims 21, 22, 45, and 46 each contain similar features and are not anticipated by Chow for at least the same reasons as claim 9.

New claims 48 and 49 contain similar features to claim 1 and are patentable over Chow for at least the same reasons as claim 1. Additionally, claim 48 is not anticipated by Chow because Chow does not disclose "adding a tracer for said first method." Similarly, claim 49 is not anticipated by Chow because Chow does not disclose "adding a timer for said first method." Therefore, Applicants respectfully assert that claims 48 and 49 are patentable over Chow.

Applicants respectfully request consideration of claims 48 and 49.

Claim 12 is not anticipated by Chow because Chow does not disclose "storing a result for a first method from an operand stack; preparing said operand stack for an invocation of a second method; invoking said second method, including providing said result to said second method; and resetting said operand stack with respect to said result to a state existing prior to said step of storing said result." The Examiner argues that these limitations are disclosed in the following passage from Chow:

> During normal execution of the bytecode program by bytecode program interpreter 31, bytecode program interpreter 31 must continually monitor the operand stack for overflows (i.e., attempting to add more data to the stack than the stack can store) and underflows (i.e., attempting to pop data off the stack when the stack is empty). Such stack monitoring must normally be performed for all bytecodes that change the status of the stack (col. 3, lines 41-49).

This passage merely states how an interpreter typically monitors an operand stack. It is unclear how the Examiner believes this passage discloses the features of claim 12. However, even using the Examiner's explanation that the "result" can be equated to an exception produced by a "first method" in Chow, Chow does not disclose that this "result" is stored "from an operand stack." Nor does Chow disclose that the "operand stack" is prepared for "an invocation of a second method" or that the "result" is provided to the "second method" when the "second method" is invoked. Therefore, claim 12 is not anticipated by Chow because Chow does not disclose any of the limitations of claim 12. Applicants respectfully assert that claim 12 is patentable over the cited prior art. Claims 4, 5, 13-17, 20-22, 25-27, 32, 38 and 42-46 each recite similar features and are patentable over Chow for at least the same reasons as claim 12. Applicants respectfully request reconsideration of claims 4, 5, 12-17, 20-22, 25-27, 32, 38 and 42-46.

II. Rejection of Claims 7, 10, 11, 19, 23, 24, 28, 35, 40, and 47 Under 35 U.S.C. §103(a)

Claims 7, 10, 11, 19, 23, 24, 28, 35, 40, and 47 have been rejected under 35 U.S.C. §103(a) as being unpatentable over Chow in view of Kukol (US 5,628,016). Applicants respectfully assert that claims 7, 10, 11, 19, 23, 24, 28, 35, 40, and 47 are not obvious over Chow

in view of Kukol because the cited prior art, alone or in combination, does not disclose, teach, or suggest all of the limitations of the rejected claims.

As discussed above, Chow does not disclose, teach, or suggest "adding new code to said existing object code including adding object code for an additional method that adds an additional function and object code that provides said result to said additional method, said object code that provides said result to said additional method is added to said first method," as recited in claim 1. Claims 7, 10, 11, 35, and 40 each contain a similar feature. Furthermore, it would not be obvious to one of ordinary skill in the art to modify Chow using the technology disclosed in Kukol to incorporate these features. Kukol discloses registering exception handlers with a computer's operating system so that when an exception occurs, an exception handler can easily be found using the stored exception registration records. Kukol does not disclose, teach, or suggest "adding new code to said existing object code." Therefore, one of ordinary skill in the art would not use Kukol to modify Chow to include these features. Applicants respectfully assert that claims 7, 10, 11, 35, and 40 are patentable over the cited prior art for at least these reasons. Reconsideration of claims 7, 10, 11, 35, and 40 is respectfully requested.

Additionally, as discussed above, Chow does not disclose, teach, or suggest "storing a result for a first method from an operand stack; preparing said operand stack for an invocation of a second method; invoking said second method, including providing said result to said second method; and resetting said operand stack with respect to said result to a state existing prior to said step of storing said result," as recited in claim 12. Claims 19, 23, 23, 28, and 47 each contain a similar feature. Furthermore, it would not be obvious to one of ordinary skill in the art to modify Chow using the technology disclosed in Kukol to incorporate these features. As discussed above, Kukol discloses registering exception handlers with an operating system. Kukol does not disclose, teach, or suggest any of the features of claim 12. Therefore, one of ordinary skill in the art would not use Kukol to modify Chow to include the features of claims 19, 23, 23, 28, and 47. Applicants respectfully assert that claims 19, 23, 23, 28, and 47 are patentable over the cited prior art for at least these reasons. Reconsideration of claims 19, 23, 23, 28, and 47 is respectfully requested.

Based on the above amendments and these remarks, reconsideration of claims 1-49 is respectfully requested.

The Examiner's prompt attention to this matter is greatly appreciated. Should further questions remain, the Examiner is invited to contact the undersigned agent by telephone.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: March 14, 2008

By: /Michelle Esteban /
Michelle Esteban
Reg. No. 59,880

VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105-4206
Telephone: (415) 369-9660
Facsimile: (415) 369-9665